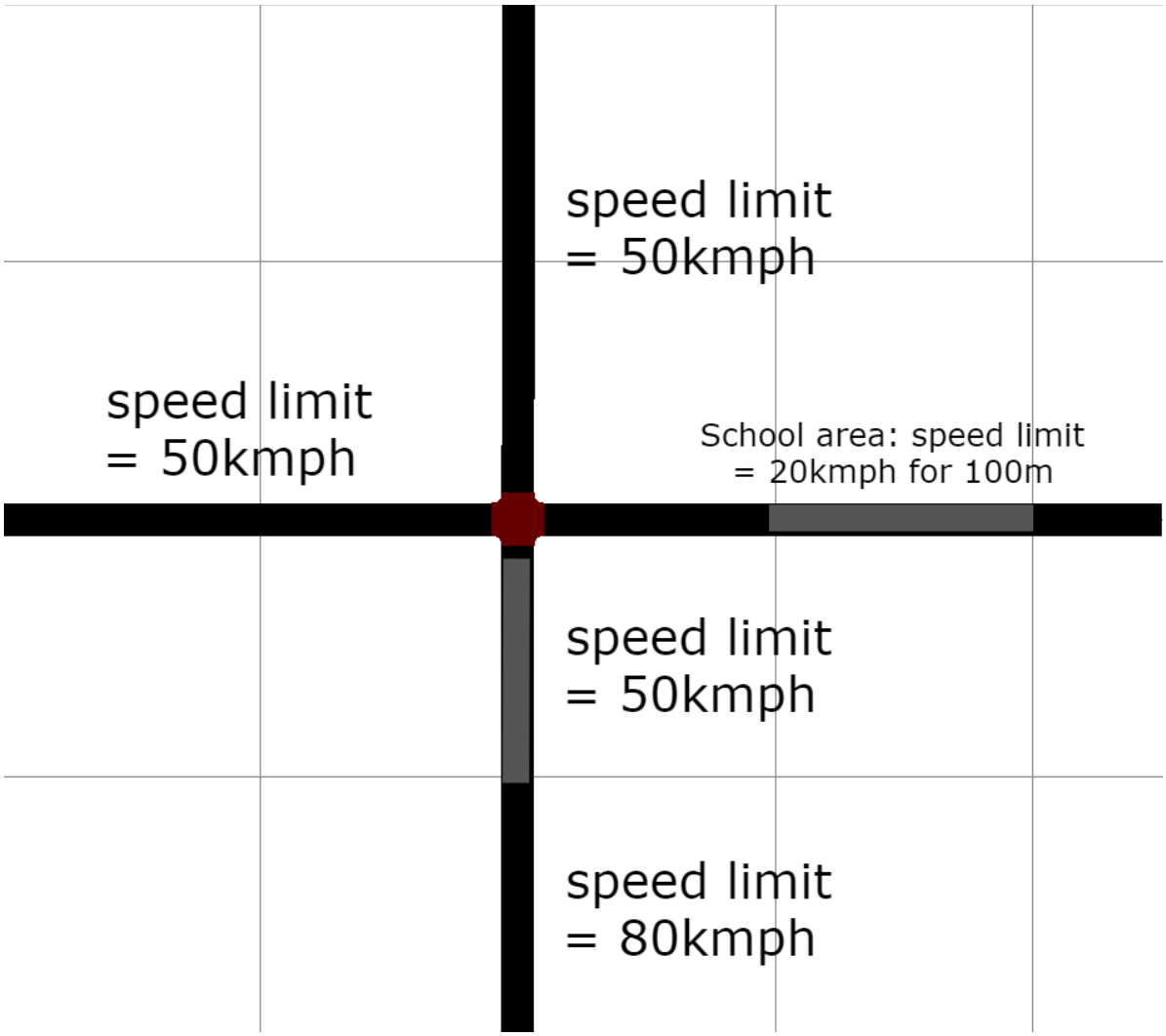
Weekly Updates – Vinayak Gajendra Panchal

PART-1: Driver type Classification

Simulation setup:

Step-length	0.10 sec (10 cycles/sec)
Collision.action	remove
Collision.stoptime	20 sec
Collision.check-junctions	True
'- - log' (simulation log files)	simulation_main.log
Number of Vehicles	200

2-Lane 4-Way Junction Details:



I have incorporated lanes with speed restrictions and a designated school zone area with limits ranging from 20-30 kmph (grey highlighted area). These speed restrictions have been implemented to enhance the replication of traffic scenarios within the LaneLet2 framework.

The tables below represent the route distribution and vehicle distribution for the simulation.

Route id	edges	probability
r_0	-E6 -E5 -E1 E3 E7	0.2
r_1	E0 E1 E5 E6	0.2
r_2	E2 E3 E7	0.2
r_3	-E7 -E3 -E0	0.2
r_4	-E6 -E5 -E1 -E0	0.2

Parameter	normal	speeder	tailgater	aggressive	violator
guiShape	passenger	passenger	passenger	passenger	passenger
carFollowModel	EIDM	EIDM	EIDM	EIDM	EIDM
color	blue	yellow	orange	red	black
probability	0.2	0.2	0.2	0.2	0.2
tau	1.5	1.5	0.1	0.1	0.1
minGap	2.5	2.5	1	1	1
accel	2.6	5	3	5	3
decel	4.5	6	5	5	5
sigmaerror	0.1	0.1	0.5	0.5	0.5
maxSpeed	33.33	41.67	33.33	41.67	33.33
emergencyDecel	4.5	6	7	7	7
jmDriveRedSpeed	-	-	-	10	-
jmDriveAfterRedTime	-	-	-	4	-
jmDriveAfterYellowTime	-	-	-	4	4
lcCooperative	-	-	-	-1	-
lcSpeedGain	-	-	-	10	-

Utilizing the specified intersection network configuration and vehicle-route distribution with speed limits, simulations were conducted for 200 vehicles, resulting in a comprehensive dataset of 130,000 records of vehicle data. This dataset was further enriched by incorporating emissions data for each vehicle over time, expanding the dataset to include 27 distinct attributes. The objective is to categorize the driving style into one of five classes: 'aggressive', 'normal', 'speeder', 'tailgater', or

'violator'. To achieve this classification, a range of supervised machine learning and deep learning models were employed and evaluated.

For the study, a variety of Machine Learning Models were explored. Each model was tested using numerical data, and categorical data that was processed with either one-hot encoding or label encoding to observe the computational demands and complexity variations. Techniques for feature reduction and selection were applied, including the ANOVA F-test for numerical features, the chi-square test for categorical features, analysis of variable correlations, and Recursive Feature Elimination with cross-validation (RFECV). Additionally, Principal Component Analysis was utilized for dimensionality reduction to evaluate its effectiveness against RFECV in terms of performance. Grid search cross-validation and Optuna libraries were used for hyperparameter tuning of these models.

Machine learning Models tested: (2 cases: one-hot and label encoding of features)

1. Multinomial Naïve Bayes
2. Gaussian Naïve Bayes
3. K-nearest neighbor (KNN)
4. Decision Tree Classifier
5. Random Forest Classifier

Deep learning models:

1. Multi-layer perceptron (Light-weight)
2. Multi-layer perceptron (Heavy)
3. Convolution Neural Network (Conv1D)

Further details of the models are in the 3 Python files provided.

Results and Discussion:

1. **Label-encoding categorical variables and numerical data**

A. All features were considered after ANOVA and chi-square test.

The Decision Tree and Random Forest models do the best job, with both scoring above 95% in accuracy and very close in precision, recall, and F1 Score, which measure how accurate and consistent they are. The KNN model also does pretty well with an accuracy of about 85%. The two Naive Bayes models have the lowest scores, with the Gaussian version doing a bit better than the Multinomial, but both are under 40% accuracy, which means they are not as good at making correct predictions for driver type.

Model	**Test Set Accuracy (%)**	**Precision**	**Recall**	**F1 Score**
Multinomial Naive Bayes	28.02	0.2872	0.2466	0.1933
Gaussian Naive Bayes	37.06	0.3642	0.3503	0.3414
KNN	84.96	0.8466	0.8476	0.8457

Decision Tree Classifier	95.5	0.9541	0.9546	0.9543
Random Forest Classifier	95.68	0.9557	0.9568	0.9561

B. Features Selection Comparison of Tree Models

Model	**Feature Selection**	**Test Set Accuracy (%)**	**Precision**	**Recall**	**F1 Score**
Decision Tree Classifier	All Features	95.5	0.9541	0.9546	0.9543
Random Forest Classifier	All Features	95.68	0.9557	0.9568	0.9561
Decision Tree Classifier	RFECV Features	97.27	0.9724	0.9724	0.9724
Random Forest Classifier	RFECV Features	97.88	0.9783	0.9787	0.9785

The table compares two types of tree-based models using different feature selection methods to see which is better at predicting outcomes. When using all available features, the Decision Tree and Random Forest models are almost equally good, with accuracy just above 95%. However, when we use a special method called RFECV to pick the best features, both models perform even better. The Random Forest with RFECV-selected features is the top performer, with an accuracy of nearly 98%. This shows that using RFECV to choose features can make models more accurate at making predictions.

2. One-hot encoding categorical variables and numerical data

A. All categories were considered after ANOVA and chi-square test.

The Decision Tree Classifier leads with the highest accuracy of 97.38% and the best F1 Score, showing it predicts great. The KNN model is also strong, with an accuracy of 86.38%. However, both Naive Bayes models don't do as well, with just over 33% accuracy, meaning they may not be the best choice here. The Random Forest Classifier is also a good model with an accuracy of 93.48%, almost matching the Decision Tree's numbers.

Model	**Test Set Accuracy (%)**	**Precision**	**Recall**	**F1 Score**
Multinomial Naive Bayes	33.59	0.3258	0.3191	0.3114
Gaussian Naive Bayes	33.74	0.3565	0.3188	0.2851
KNN	86.38	0.8610	0.8619	0.8603
Decision Tree Classifier	97.38	0.9735	0.9733	0.9734
Random Forest Classifier	93.48	0.9333	0.9343	0.9334

B. Dimensionality reduction RFCV and PCA

The table compares how different machine learning models did when dimensionality was reduced using: PCA and RFECV. PCA didn't work so well for the Multinomial Naive Bayes model, which had the lowest scores. The KNN, Decision Tree, and Random Forest models did much better with PCA, all scoring over 80% in accuracy. However, when the Decision Tree and Random Forest models used the RFECV method, which picks the most important data to use, their scores went up even higher, with the Random Forest model getting the best score of almost 99% accuracy. This tells us that RFECV can really help these models make better predictions.

Model	**Feature Selection**	**Test Set Accuracy (%)**	**Precision**	**Recall**	**F1 Score**
Multinomial Naive Bayes (PCA)	PCA	24.89	0.1337	0.2037	0.0892
Gaussian Naive Bayes (PCA)	PCA	37.24	0.3636	0.3635	0.3486
KNN (PCA)	PCA	83.05	0.8270	0.8279	0.8266
Decision Tree Classifier (PCA)	PCA	94.67	0.9461	0.9461	0.9461
Random Forest Classifier (PCA)	PCA	94.98	0.9483	0.9491	0.9486
Decision Tree Classifier (RFECV)	RFECV	97.04	0.9701	0.9695	0.9698
Random Forest Classifier (RFECV)	RFECV	98.79	0.9876	0.9878	0.9877

3. Deep Learning Models

The table displays performance metrics for three different deep-learning models. The MLP1 (LIGHT) model, with the least number of parameters at 19,461, achieved an accuracy of 83.01%, precision of 0.8314, and F1 score of 0.8283 after 368.127 seconds of training. The MLP2 (HEAVY) model, which is more complex with 470,021 parameters, outperformed MLP1 with higher accuracy (89.86%), precision (0.9012), and F1 score (0.8978), and it also trained faster, taking only 289.627 seconds. The CNN model, despite having a moderate number of parameters at 291,781, had a lower performance with an accuracy of 79.92% and an F1 score of 0.7962, and it took the longest to train at 567.413 seconds. These results illustrate the balance between model complexity, training time, and performance accuracy, suggesting that MLP2 (HEAVY) offers the best performance among the three models for the given task.

Model	**Training Time (sec)**	**Test Loss**	**Test Accuracy**	**Precision**	**Recall**	**F1 Score**
MLP1 (LIGHT)	368.127	0.4284	83.01%	0.8314	0.8302	0.8283
MLP2 (HEAVY)	289.627	0.2422	89.86%	0.9012	0.8956	0.8978

CNN	567.413	0.5064	79.92%	0.7972	0.8002	0.7962

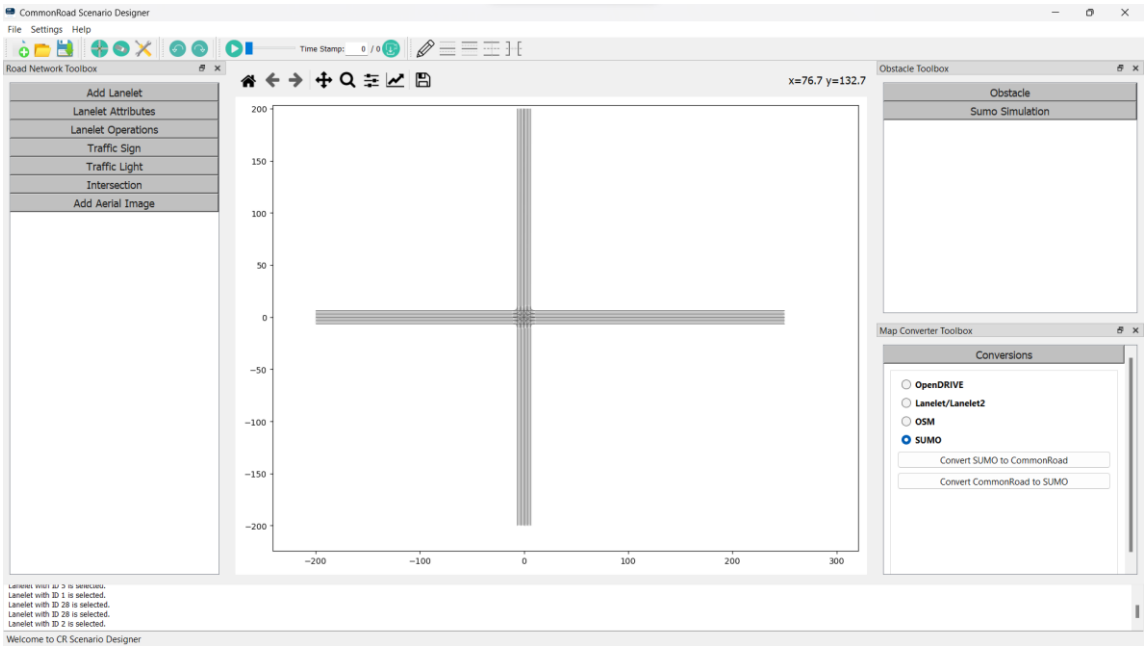
Overall Comparison: For traditional ML models, using all features post-ANOVA and chi-square testing, the Decision Tree and Random Forest classifiers significantly outperform the Naive Bayes and KNN models, boasting accuracies above 95%. However, when employing feature selection via RFECV, these tree-based models' performance is even more impressive, with the Random Forest Classifier nearly hitting a perfect 99% accuracy.

In contrast, when deep learning models are considered, the MLP2 (HEAVY) model, despite its complexity, not only achieves higher accuracy than its lighter counterpart and the CNN model but also trains more quickly. This suggests that while more complex models require more computational resources, they can indeed provide better results if tuned correctly. The CNN, despite its longer training time and moderate parameter count, lags slightly behind in accuracy.

PART 2: SUMO to LaneLet2 format.

To convert SUMO net-edit file to LaneLet2 osm file, we used a library and gui tool named crdesigner (Common Road Scenario Designer)https://gitlab.lrz.de/tum-cps/commonroad-scenario-designer

Below is the osm file opened in the crdesigner gui.



Next steps will be to verify the Lanelet2 osm file's attributes and verify if all the information related to lanelets and regulatory rules are present in the osm file.